

## **CLAIM AMENDMENTS**

### **Claim Amendment Summary**

#### **Claims pending**

- Before this Amendment: Claims 1-28, 34-42, 45 and 46.
- After this Amendment: Claims 1-26, 34-42, 45 and 46.

**Non-Elected, Canceled, or Withdrawn claims:** None.

**Amended claims:** 28, 34, and 36.

**New claims:** None.

---

#### **Claims:**

- 1. (Previously Presented)** One or more computer-readable media having stored thereon computer-executable instructions of implementing a kernel emulator for non-native program modules that, when executed by one or more processors, causes the one or more processors to perform actions comprising:
  - intercepting non-native kernel calls from non-native program modules, the non-native kernel calls calling a native kernel having access to hardware through one or more device drivers and hardware interfaces native to the native kernel;
  - converting the intercepted non-native kernel calls into native kernel calls; and
  - delivering the converted native kernel calls to the native kernel without the non-native program modules being modified to target a native platform running the native kernel on which the non-native program modules are not designed to run, thereby

facilitating interoperability of the non-native program modules within the native platform.

**2. (Previously Presented)** One or more computer-readable media as recited in claim 1, wherein the converting further comprises translating a non-native paradigm for passing parameters into a native paradigm for passing parameters.

**3. (Previously Presented)** One or more computer-readable media as recited in claim 1, wherein the converting further comprises translating non-native CPU instructions into native CPU instructions.

**4. (Previously Presented)** One or more computer-readable media as recited in claim 1, wherein the converting further comprises translating addresses from non-native length into native length.

**5. (Previously Presented)** One or more computer-readable media as recited in claim 1, wherein the converting further comprises converting non-native argument format into native argument format.

**6. (Previously Presented)** One or more computer-readable media as recited in claim 1, wherein the converting further comprises translating words from non-native word size into native word size.

**7. (Previously Presented)** One or more computer-readable media as recited in claim 1, wherein the kernel emulator further comprises limiting addressable memory to a range addressable by non-native program modules.

**8. (Previously Presented)** One or more computer-readable media as recited in claim 1, wherein the kernel emulator further comprises managing memory space that is accessible to both native and non-native program modules.

**9. (Previously Presented)** One or more computer-readable media as recited in claim 1, wherein the kernel emulator further comprises synchronizing a native shared data structure with a non-native shared data structure.

**10. (Previously Presented)** One or more computer-readable media as recited in claim 1, wherein the kernel emulator further comprises:

managing memory space accessible to both native and non-native program modules; and

mapping versions of process shared data structures (process SDSs) and versions of thread shared data structures (thread SDSs) between native and the non-native program modules.

**11. (Previously Presented)** An operating system on the one or more computer-readable media, comprising:

a native kernel configured to receive calls from native program modules; and  
a kernel emulator as recited in claim 1 configured to receive and convert calls from non-native program modules for direct handling by the native kernel without the non-native program modules being modified to natively call the native kernel.

**12. (Previously Presented)** An operating system on the one or more computer-readable media, comprising:

a native kernel configured to receive calls from native APIs;  
a kernel emulator as recited in claim 1 configured to receive calls from non-native APIs for direct execution by the native APIs without the non-native APIs being modified to natively utilize the native APIs.

**13. (Previously Presented)** A method of emulating a kernel for non-native program modules, the method comprising:

intercepting non-native kernel calls from non-native program modules, the non-native kernel calls calling a native kernel having access to hardware through one or more device drivers and hardware interfaces native to the native kernel;  
converting the intercepted non-native kernel calls into native kernel calls; and

delivering the converted native kernel calls to the native kernel without the non-native program modules being modified to target native platform running the native kernel on which the non-native program modules are not designed to run.

**14. (Original)** A method as recited in claim 13, wherein the converting step comprises translating a non-native paradigm for passing parameters into a native paradigm for passing parameters.

**15. (Original)** A method as recited in claim 13, wherein the converting step comprises translating non-native CPU instructions into native CPU instructions.

**16. (Original)** A method as recited in claim 13, wherein the converting step comprises translating addresses from non-native length into native length.

**17. (Original)** A method as recited in claim 13, wherein the converting step comprises translating words from non-native word size into native word size.

**18. (Original)** A method as recited in claim 13 further comprising limiting addressable memory to a range addressable by non-native program modules.

**19. (Original)** A method as recited in claim 13 further comprising synchronizing a native shared data structure with a non-native shared data structure.

**20. (Original)** A method as recited in claim 13 further comprising mapping versions of process shared data structures (SDSs) between native and non-native program modules.

**21. (Original)** A method as recited in claim 20, wherein a process SDS of a native program module includes a pointer to a process SDS of a non-native program module.

**22. (Original)** A method as recited in claim 20, wherein a process SDS of a non-native program module includes a pointer to a process SDS of a native program module.

**23. (Original)** A method as recited in claim 13 further comprising mapping versions of thread shared data structures (SDSs) data structure between native and non-native program modules.

**24. (Original)** A method as recited in claim 23, wherein a thread SDS of a native program module includes a pointer to a thread SDS of a non-native program module.

**25. (Original)** A method as recited in claim 23, wherein a thread SDS of a non-native program module includes a pointer to a thread SDS of a native program module.

**26. (Currently amended)** A computer comprising:  
one or more processors; and  
memory coupled to the one or more processors, the memory storing thereon computer-executable instructions that, when executed by the one or more processors, perform a method comprising:

intercepting non-native kernel calls from non-native program modules, the non-native kernel calls calling a native kernel having access to hardware through one or more device drivers and hardware interfaces native to the native kernel;

converting the intercepted non-native kernel calls into native kernel calls;  
and

delivering the converted native kernel calls to the native kernel without the non-native program modules being modified to target native platform running the native kernel on which the non-native program modules are not designed to run  
the method as recited in claim 13.

27-33 (Cancelled).

**34. (Currently amended)** A method comprising:

emulating a non-native kernel for a native computing platform by converting non-native kernel calls calling a native kernel from non-native applications into native kernel calls to the native kernel, without the non-native applications being modified to target [[he]] the native computing platform on which the non-native applications are not designed to run.

**35. (Previously Presented)** A method as recited in claim 34, wherein the emulating step further comprises:

translating non-native CPU instructions into native CPU instructions;  
translating addresses from non-native length into native length;  
limiting addressable memory to a range addressable by non-native program modules.

**36. (Original)** A method as recited in claim 35, wherein the emulating step further comprises translating a non-native paradigm for passing parameters into a native paradigm for passing parameters.

**37. (Original)** A method as recited in claim 34, wherein the converting step further comprises translating words from non-native word size into native word size.

**38. (Currently amended)** A computer comprising one or more computer-readable media having computer-executable instructions that, when executed by the

computer, perform a method comprising emulating a non-native kernel for a native computing platform by converting non-native kernel calls calling a native kernel from non-native applications into native kernel calls to the native kernel without the non-native applications being modified to target the native computing platform on which the non-native applications are not designed to run the method as recited in claim 34.

**39. (Previously Presented)** A computer-readable medium having computer-executable instructions that, when executed by a computer, emulates a non-native kernel for a native computing platform by converting non-native kernel calls calling a native kernel from non-native applications into native kernel calls without the non-native applications being modified to target on the native computing platform on which the non-native applications are not designed to run.

**40. (Previously Presented)** One or more computer-readable media having stored thereon instructions implementing a kernel emulator for non-native program modules, the instructions, when executed by a computing device, causing the computing device to emulate a non-native kernel for a native computing platform so that non-native kernel calls that call a native kernel from non-native applications are converted into native kernel calls to the native kernel without the non-native applications being modified to target on the native computing platform on which the non-native applications are not designed to run.

**41. (Previously Presented)** One or more computer-readable media having stored thereon instructions implementing the kernel emulator recited in claim 40, wherein the instructions of implementing the kernel emulator comprises:

instructions implementing an instruction-translator configured to translate non-native CPU instructions into native CPU instructions;

instructions implementing an address-translator configured to translate addresses from non-native length into native length; and

instructions implementing a memory constrainer configured to limit addressable memory to a range addressable by non-native program modules.

**42. (Previously Presented)** One or more computer-readable media having stored thereon instructions of an operating system that, when executed on a computing device, cause the computing device to implement a plurality of modules, the instructions comprising:

instructions of implementing a native kernel configured to receive calls from native program modules;

instructions of implementing a kernel emulator as recited in claim 40 configured to receive calls from non-native program modules.

**43-44. (Canceled).**

**45. (Previously Presented)** One or more computer-readable media having stored thereon instructions that, when executed by a computing device, causes the computing device to implement a kernel emulator for non-native program modules, the kernel emulator comprising:

an interceptor configured to intercept non-native kernel calls that call a native kernel from non-native program modules, the native kernel being software that operates system functions;

a call-converter configured to convert the non-native kernel calls intercepted by the interceptor into native kernel calls, wherein the call-converter comprises:

an instruction-translator configured to translate non-native CPU instructions into native CPU instructions;

an address-translator configured to translate addresses from non-native length into native length; and

an I/O unit configured to deliver converted native kernel calls to the native kernel, wherein the call-converter enables the non-native program modules to call the native kernel without the non-native program modules being modified to target platform running the native kernel for which the non-native program modules are not designed.

**46. (Original)** An operating system on a computer-readable medium, comprising:

a native kernel configured to receive calls from native program modules;

a kernel emulator as recited in claim 45 configured to receive calls from non-native program modules.

**47-50. (Cancelled).**